ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Relazione Progetto Reti
Traccia 1 – IoT

Nicola Costa - 0000922493
Rostyslav Dovganyuk - 0000923292

Giugno 2021

Introduzione

Il progetto d'esame consiste in uno scenario di Smart Meter IoT in cui 4 rilevatori di temperatura e umidità del terreno, detti Device, si collegano una volta al giorno, con una connessione UDP, verso il Gateway ed inviano le misure che in quel lasso di tempo hanno raccolto. Le misure consistono in ora della misura, temperatura e umidità e vengono salvate in un file, uno per device.
Successivamente, una volta che i pacchetti di tutti i dispositivi sono arrivati al Gateway, questo instaura una connessione TCP verso un server Cloud dove i valori vengono visualizzati sulla sua console nel seguente modo: **ip_address_device_1 – ora_misura – valore_temperatura – valore_umidità**.
Di seguito una rappresentazione della situazione, con tanto di indirizzamenti a cui devono appartenere i soggetti:

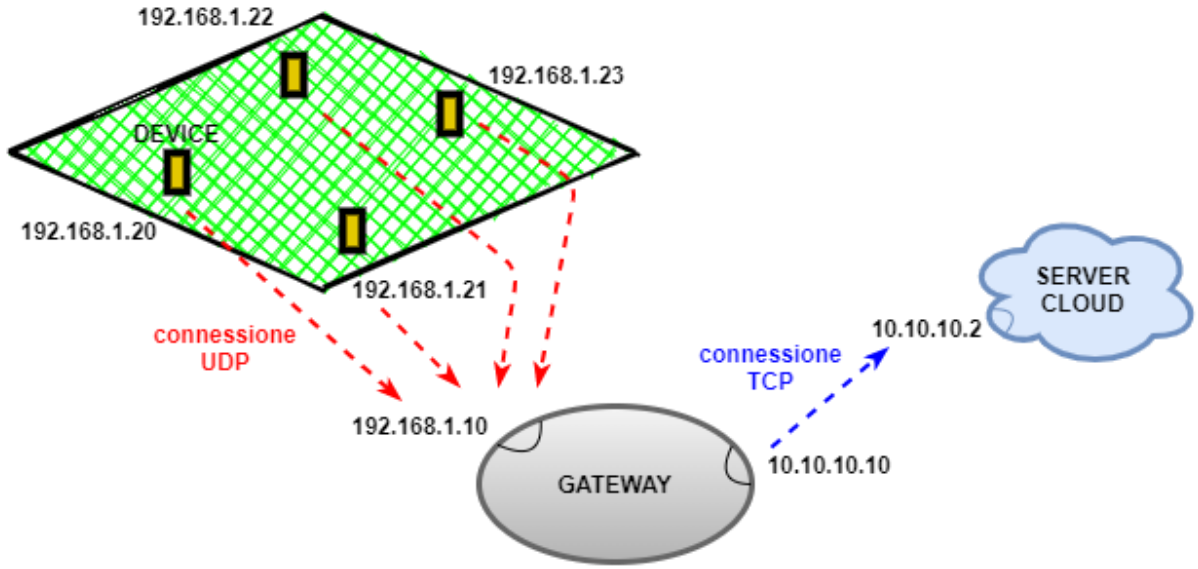

FIGURA 1

Per l'implementazione in ambiente Python sono stati creati 4 file: Device.py, IstDevice.py, Gateway.py, Cloud.py.
Il sorgente di Device.py contiene la classe Device e i metodi necessari al suo funzionamento, mentre quello di IstDevice.py istanzia i quattro device sottoforma di thread concorrenti.
Gateway.py e Cloud.py contengono rispettivamente i sorgenti per il funzionamento del Gateway e del server Cloud.
Per la corretta operatività dell'intero sistema è necessario **avviare per primo il server Cloud**, **per secondo il Gateway** e **per ultimi i device**.

Device

Il device esegue le misurazioni di temperatura e umidità del terreno e le memorizza in un file, dopo le 24 ore si connette al Gateway e gli invia le misurazioni.
Per simulare al meglio il funzionamento del device, si è scelto di strutturarlo nel seguente modo:

1. Le tempistiche:
   per evitare di aspettare veramente 24 ore, viene utilizzato un timer che risveglia il device per fare le misurazioni, ripetitivamente *n* volte (*n* è la costante RANGE_RILEVATION nel codice).
2. Le misurazioni e il salvataggio su file:
   le rilevazioni della temperatura e dell'umidità vengono generate randomicamente e successivamente salvate con l'orario della rilevazione in un file di testo con nome "device" + ultima parte dell'IP + ".txt", con l'aggiunta dell'orario per ogni rilevazione.

```
23  #metodo per il salvataggio dei dati nel file
24  def saveOnFile(ip,hour,temp,umid):
25      #split dell'ip per nominare il file in modo esplicativo "device20.txt"
26      a,b,c,d = ip.split('.')
27      file = open(f'device{d}.txt','a')
28      string = f'{hour} - {temp} - {umid}\n'
29      file.write(string)
30      file.close()
```

split dell'indirizzo IP device e salvataggio dati nel file

3. Invio delle misurazioni:
   dopo aver raccolto le misurazioni, nel nostro caso 24, il device apre un socket UDP verso il Gateway sulla porta 10000 e invia il messaggio contenente tutte le rilevazioni estratte dal file con l'aggiunta del suo indirizzo IP e del tempo di inizio spedizione pacchetti UDP. **Vedere punto 1 FIGURA 3.**

Da progetto sono richiesti 4 dispositivi, quindi è stata creata la classe Device all'interno di Device.py ed il file IstDevice.py si occupa di istanziarla passandogli come argomenti l'indirizzo IP ed il timer, per poi successivamente farli partire in thread diversi simulando un funzionamento indipendente e parallelo. Nel caso si vogliano aggiungere più dispositivi basta inserire nella lista ip_address_devices, presente in IstDevice.py, altri indirizzi IP, cosicché il programma creerà in automatico i dispositivi.

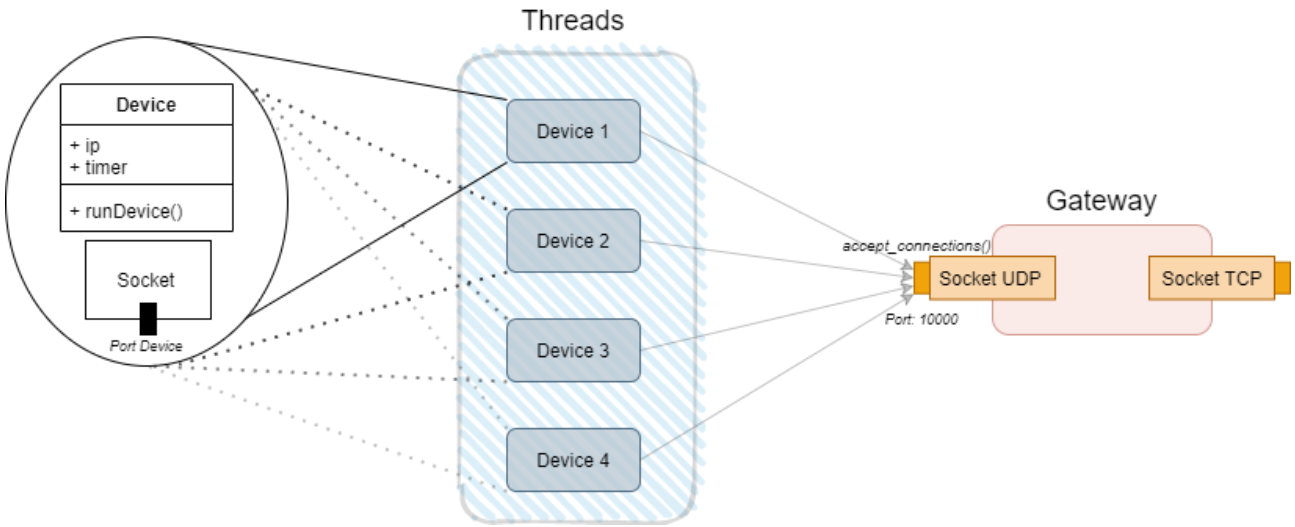

FIGURA 2 - SCHEMA THREAD

Gateway

Le due funzioni principali svolte dal Gateway sono:

- restare in ascolto e aspettare che tutti i device inviino le misurazioni;
- elaborare i dati ed inviarli al server Cloud.

Il Gateway quando viene acceso apre subito una connessione di "ascolto" sulla porta 10000, di tipo UDP, che resterà sempre aperta finché non viene dato lo stop dall'utente; questa interfaccia è aperta sulla stessa rete dei device e serve per ricevere le misurazioni.
Per controllare che le misurazioni ricevute siano appartenenti a tutti i corretti device, tutte le volte che il Gateway riceve dei messaggi, li spacchetta, splittando la stringa (come visto in precedenza alla stringa delle misurazioni viene aggiunto anche indirizzo IP e tempo di inizio invio pacchetti UDP) e ricavando l'IP del device e il tempo iniziale di invio, salvando poi in un dizionario IP → rilevazioni e calcolando poi il tempo totale di trasmissione del pacchetto UDP.
Ogni volta che viene aggiunto un elemento nuovo al dizionario, si controlla se il set di chiavi del dizionario, gli IP dei device in questo caso, sia uguale al set degli IP conosciuto dal gateway e, nel caso siano uguali, viene preparato il messaggio come richiesto da progetto (*IP_address_device_x – ora_misura – valore_temperatura – valore_umidità*). **Vedere punto 2 FIGURA 3.**

Viene quindi creato il socket con connessione TCP collegandolo allo stesso indirizzo e porta (127.0.0.1, 9000) ai quali il server Cloud è in ascolto e inviato il messaggio con i dati. Ricevuta una response si procede a rilasciare il canale TCP.

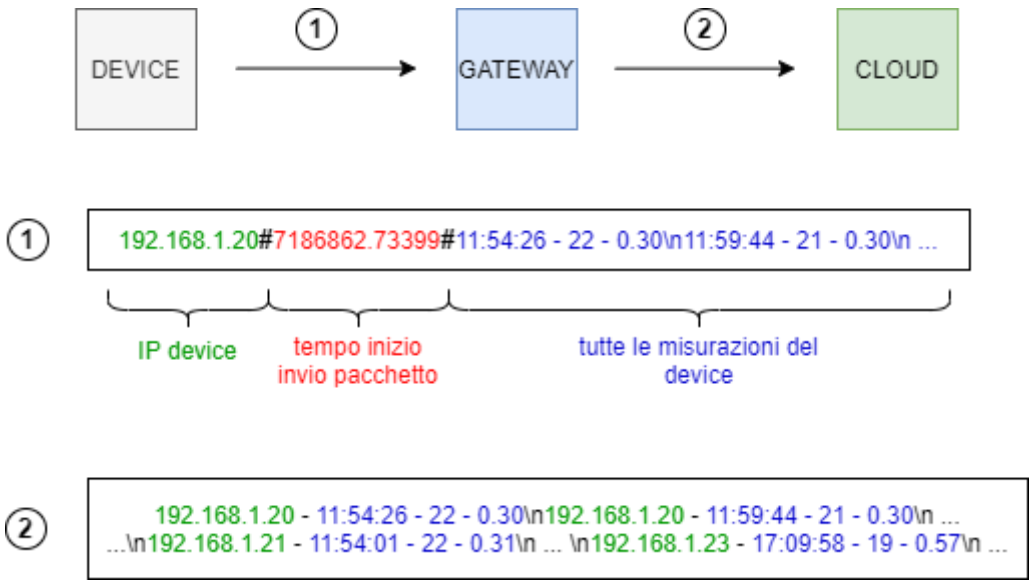

FIGURA 3 - SCHEMA DEL MESSAGGIO INVIATO

Cloud

Il server in questione si occupa solamente di stampare il messaggio, contenente i dati già elaborati come da requisito del progetto, sulla propria console. Per farlo si connette al Gateway implementando un socket con connessione TCP, in ascolto sulla porta 9000 all'indirizzo 127.0.0.1 (localhost). Non appena il server accetta la richiesta di connessione con il Gateway, parte il timer per calcolare il tempo di trasmissione di tutti i pacchetti, che termina, appunto, quando tutti i pacchetti del messaggio arrivano a destinazione. Dopo aver stampato decodificando il messaggio, invia una risposta affermativa di arrivo dei pacchetti (nel caso non ci siano stati errori) e rilascia il canale TCP.

Strutture dati utilizzate

Le sole strutture dati utilizzate sono state **dictionary** e **set**.
Il dizionario usato è dictionary_data, con chiave l'IP del device e valore la stringa con le misurazioni di quel device, è stato molto comodo da manipolare e il più giusto da utilizzare per il contesto che si voleva implementare e per immagazzinare in un'unica struttura dati sia gli IP, usati poi per il controllo con i set, sia le misurazioni.
I set sono stati utili per controllare se gli IP dei device che hanno inviato i loro dati fossero tutti e 4, confrontandoli con gli indirizzi IP che il Gateway conosce. Visto che i set contengono oggetti senza mantenere l'ordine al loro interno è stata la soluzione giusta dato che si stava lavorando con dei thread concorrenti (il primo che parte non è detto che sia il primo a finire, così per il secondo, il terzo...).

```
67      #controllo di aver ricevuto le misurazioni da tutti i client
68      if set(dictionary_data.keys()) == ip_address_devices:
69          #creo le stringhe come richieste da progetto
70          message = create_message_to_server(dictionary_data)
71          print('Received all devices misurements')
```

controllo con i set

Buffer per la ricezione dei messaggi

Le dimensioni scelte per i buffer di ricezione sono:
Socket UDP Gateway: **2048** perché riesce a contenere correttamente i Byte che si vogliono inviare e rende scalabile il programma: ad esempio con più misurazioni nell'arco delle 24 ore rispetto al numero attuale.
Socket TCP server Cloud: **4096**, sicuramente maggiore della precedente vista la mole di Byte circa 4 volte superiore.

Librerie utilizzate

- socket
- threading
- time
- sys
- signal
- random
- datetime

Link repository GitHub: https://github.com/rostydovga/Programmazione-Reti